Performance Impact of Misbehaving Voters

Mohammed Alotaibi[1] and Nigel Thomas[1]

Newcastle University, Newcastle upon Tyne, UK
{m.alotaibi1,nigel.thomas}@newcastle.ac.uk

Abstract. In this paper we present three formal performance models, using PEPA, for three types of misbehaving voters when using the DRE-i e-voting system. We use the constructed performance models to study the impact of the intervention of misbehaving voters on the throughput of four main actions of the DRE-i e-voting system. Our performance analysis reveals that the three types of misbehaving voters have a negative impact on the throughput of the DRE-i server actions.

Keywords: Performance Models · PEPA · e-voting.

1 Introduction

E-voting systems face a wide range of potential misbehaving components or agents beyond what we used to have in traditional elections. These misbehaviours include misconfigured e-voting components [23, 15], errors made by voters, and malicious behaviours made by attackers [16]. Some known attacks on e-voting systems include the replay attack [6], man in the middle attacks, and Cross-Site Scripting (XSS) and Cross-Site Request Forging (CSRF) attacks [7, 19]. Investigating the impact of these misbehaviours on e-voting systems performance is an intriguing research topic. One way of studying this impact is through constructing the misbehaving voters' formal performance models and evaluating how their interventions with e-voting system may affect the performance of the e-voting system.

PEPA (Performance Evaluation Process Algebra) [13] is a well-known formalism in constructing performance models for concurrent systems and communication protocols [9, 22]. In [2], a formal performance model for the casting-verifying stage of the DRE-i e-voting scheme [11] was constructed using PEPA. In this study, we will model three types of misbehaving voters and their interactions with the DRE-i system using the same formalism. The constructed performance models will be analysed using performance evaluation techniques built in PEPA Eclipse Plug-in [21] to have an insight on the effect of misbehaving voters on the performance of the DRE-i server.

Next, we will provide a brief background about e-voting systems and PEPA. In the third section, we will describe our approach in modelling the misbehaving voters and analysing their impact on the DRE-i e-voting system. Performance models will be shown in section four, and in section five we will present the result and discussion. Finally, conclusion and future work will be presented in section six.

2 Background

In this section, we will provide a brief background about PEPA (For more details about PEPA, please refer to [13]) and e-Voting schemes (Refer to [10, 12] for more details). Also, we will briefly present some known misbehaviours that could affect e-voting systems. At the end of this section, we will present the work related to modelling and analysing e-voting schemes using formal performance models such as PEPA.

2.1 e-Voting

An election enables a participant to choose his candidate for holding a position in a public or private organisation by the voting process. To increase the turnout of voters in elections, researchers suggested electronic voting systems that meet strict accuracy and security requirements. Well-known examples of e-voting systems include DRE-i [11], Helios [1], and iVote [5]. Many countries, states, and organizations have used the e-voting systems in elections such as Estonia [17], Brazil, India, the Australian state of New South Wales (NSW) [5], and the International Association of Cryptologic Research (IACR)[3]. Electronic voting security literature identifies many security requirements for e-voting protocols such as completeness, privacy, soundness and robustness, receipt-freeness, verifiability, fairness, eligibility, and unreusability. To achieve these security features, the electronic voting schemes use different cryptographic building blocks which include blind signatures, mix-nets, encryption algorithms, and interactive and non-interactive proofs.

2.2 e-Voting Misbehaviours

Security of e-voting systems is very influential in the democratic process so many researchers have studied the possible attacks against e-voting systems [12, 16]. One of the attacks is the replay attack where a malicious voter retransmits a valid vote or message. The vote replay attack was discovered in different e-voting schemes such as Helios 2.0 [6] and the e-voting schemes by Sako & Kilian and Schoenmakers [18]. Another attack is based on compromising the web-interface of the e-voting client using for example XSS (Coss-Site Scripting) or CSRF (Cross-Site Request Forgery) attacks. In [7] the malicious voter can install a malicious browser extension on the voter's machine to compromise Helios 2.0 e-voting system. Using the CSRF approaches [19], the malicious voter may exploit a weakness in the e-voting web interface and establish an authenticated session with the e-voting server to exchange voting messages with the server.

2.3 Introduction to PEPA

PEPA (Performance Evaluation Process Algebra) is a stochastic modelling formalism for constructing performance models for concurrent systems [13]. It was

successfully used in modelling and analysing the performance aspects of systems and protocols. Performance models constructed by PEPA can help systems designers to evaluate the performance characteristics of the system to be deployed. The performance attributes (such as throughput, queue-length, and response time) of models constructed by PEPA can by analysed using Continuous-Time Markov Chain (CTMC) and Ordinary Differential Equations (ODEs) approaches. When using CTMC, the construction and evaluation of PEPA models will be restricted by the size and complexity of modelled systems. PEPA model will encounter the state-space explosion problem when the model comprises a large number of components. To overcome the state-space problem, a fluid approximation approach has been suggested [14] to represent the PEPA model's underlying CTMC as a set of ordinary differential equations (ODEs). PEPA is an abstract compositional description formalism which is used for constructing performance models as a number of interacting components that process activities with rates. In a PEPA model, activity rate is an exponentially distributed random number that shows the rate at which the activity (action) happens during the execution of the model. To evaluate the performance of the PEPA model, the PEPA Eclipse Plug-in tool [21] is used to edit and test the model. It is also used to derive the model's underlying CTMC and the ODE approximation of the model's CTMC. The derived CTMC and ODEs can be solved by the tool to extract the model's performance measures such as expected response time, throughput and utilisation.

The syntax of PEPA language is composed of combinators that express the behaviours and interactions of the model's components. The following is the set of PEPA language's combinators:

Prefix The prefix combinator "**.**"designates the first behaviour undertaken by the component. The action type a and rate r for component P is encoded in PEPA as $(a, r).P$ which means the action will be carried out and then behaves as component P.

Constant The constant combinator $\stackrel{\text{def}}{=}$ assigns names to behaviours (components). For example, $Q \stackrel{\text{def}}{=} (a, r).P$ represents the assignment of the behaviour of $(a, r).P$ to the component Q.

Cooperation The combinator "$\bowtie_{\mathbf{L}}$"represents the interactions between components. The $(P \bowtie_{\mathbf{L}} Q)$ indicates the cooperation between components P and Q over action types in the cooperation set L. The two components P and Q will proceed independently and concurrently when their cooperation set L is empty. In this case, the parallel composition of P and Q will be expressed as $P || Q$.

Choice The choice combinator "+"denotes the competition between behaviours. $P+Q$ represents a system that may behave as P or Q.

Hiding The hiding combinator "/"hides the activities in the set L and considers them as an internal delay inside the component. The P/L makes the activities in set L as the unknown type τ where the external observer can witness the delay caused by the hidden activity τ. However, the external observer can not access the hidden activity.

2.4 Related Work

PEPA performance models for the voting scheme of Fujioka, Okamoto, and Ohta [8] was constructed and evaluated in [20, 4].In [20], a set of PEPA models constructed and analysed for reliable and unreliable voters. In [4], a stochastic simulation technique was used to convert a PEPA model of an e-voting scheme to a set of rate equations. Each rate equation represented an individual action of a component inside the PEPA model and by using these rate equations a simulation description file was constructed that fitted the Dizzy simulation tool. Therefore, the PEPA model of the e-voting scheme was simulated and analysed for a large number of voters. Moreover, the DRE-i e-voting scheme was modelled by PEPA in [2]. The constructed model was evaluated using CTMC and ODEs approaches for a varying number of voters to evaluate the voters' response time when they get involved in the cast-verify stage during the election day.

3 Our Approach

During the election day, legitimate voters who prefer using e-voting systems will use their electronic devices to join election and cast their votes. This usually will lead to an increase in the throughput of the main activities in the e-voting system. With the intervention of misbehaving voters with the e-voting system, the throughput that is dedicated for legitimate voters will be challenged. We are interested in evaluating the impact of the rate and depth of the intervention of misbehaving voters with the e-voting system. Theretofore, in this section we will present the PEPA models for the DRE-i e-voting server and client, and the three types of misbehaving voters. Also, we will present the rates for the models' actions, and finally we will explain how to evaluate the impact of the misbehaving voters on the DRE-i e-voting system.

3.1 DRE-i Behaviour description

The Direct Recording Electronic with integrity e-voting scheme (DRE-i) was presented by Hao *et al.* [11]. The scheme is an end-to-end verifiable and self-enforcing cryptographic voting scheme based on the Direct Recording Electronic voting systems which replaces the tallying authority with a cryptographic homomorphic tallying algorithm. The DRE-i scheme can be used in controlled or uncontrolled voting environment for large-scale country-wide political elections or small-size elections like university students' union elections. In this e-voting scheme, a tamper-resistant security module of the e-voting server will generate for each eligible voter n ballots. Each ballot will have two encrypted values known as cryptograms. During election stage, the voter needs to prove his eligibility for voting and identity to the voting server. If he is eligible for voting, the voter will receive a ballot and will choose one of the two cryptograms. The voter will submit his selected vote to the server. Next, the server will sign the received ballot and send it to the voter so he can either accept it and cast it as

his vote or reveal the content of signed ballot to verify that his selection reflects his intention.

We are interested in modelling and evaluating the casting stage of the DRE-i e-voting system which has four main activities: getting the vote cryptograms, signing the selected cryptogram, casting or verifying the selected vote. From the server side, these actions are *voteCryptogramsReply*, *signTransReply*, *voteCastAck*, and *voteVerifyAck*. Figure 1 demonstrates the interactions between the voting client and voting server to carry out vote casting process.

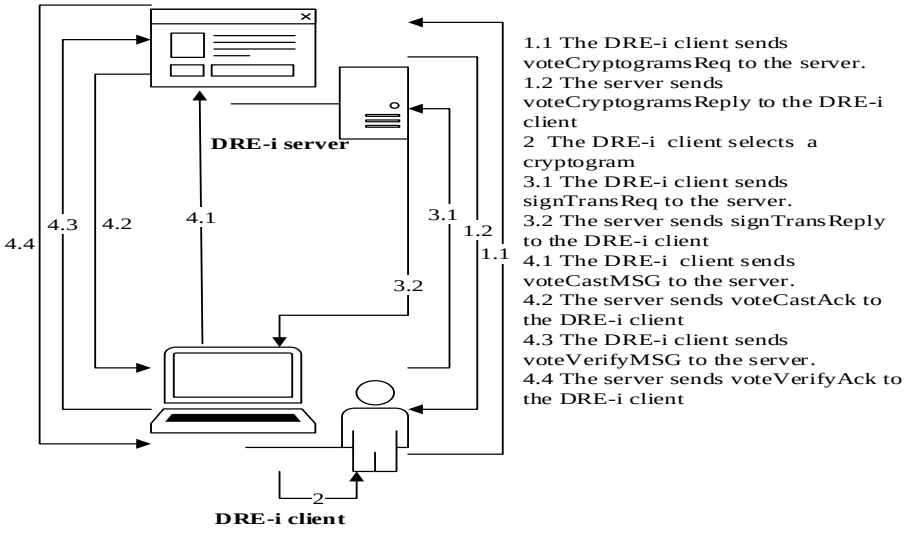

Fig. 1: Main collaboration between the *DRE-i* client and server

The legitimate voter will send the following requests: voteCryptogramsReq, signTransReq, voteCastMSG, and voteVerifyMSG to the server. Respectively, the server will reply with the following actions: voteCryptogramsReply, signTransReply, voteCastAck, and voteVerifyAck. In the following two paragraphs we will explain the server and client actions.

3.2 Misbehaving voters

We have also investigated three misbehaviours that may happen to the DRE-i e-voting system. The first misbehaviour is represented by a rogue client that just replays voteCryptogramsReq messages to get valid ballots from the DRE-i server. We call this type of rogue clients as RCA (Rogue Client of type A) voting client and the steps 1.1 and 1.2 in Fig. 1 show the interactions between the RCA voting

client and voting server. This behaviour represents an unsuccessful replay attack where the misbehaving voter tries to request a ballot that has been requested by a legitimate voter but the voting server replies with error message.

The second misbehaviour is represented by a rogue client that sends requests voteCryptogramsReq and signTransReq to the DRE-i server. We call this type of rogue clients as RCB (Rogue Client of type B) voting client and the steps 1.1, 1.2, 2, 3.1, and 3.2 in Fig. 1 show the interactions between the RCB voting client and voting server. In this type of voter misbehaviours, the voter successfully replays a ballot request and gets back the vote cryptograms from the server and then selects a candidate and sends a sign transcript request to the server but the server replies with an error message.

The third misbehaviour is represented by a rogue client that successfully sends requests voteCryptogramsReq, signTransReq, voteCastMSG, and voteVerifyMSG to the DRE-i server and gets back valid replies. We call this type of rogue clients as RCC (Rogue Client of type C) voting client and the steps 1.1, 1.2, 2, 3.1, 3.2, 4.1, 4.2, 4.3 and 4.4 in Fig. 1 show the interactions between the RCC voting client and the voting server. In this type of misbehaving voters, the malicious voter can successfully cast or verify a vote. This type of misbehaviour can be represented by cross-site scripting (XSS) or Cross-Site Request Forgery (CSRF) attacks.

3.3 Actions Rates

We consider the rogue clients' actions to have rates similar to the rates of the legitimate clients. We assumed that the rogue clients need to wait for one second (1000 ms) to restart the next intervention. The rogue clients will have different rates to complete one intervention with the DRE-i system. To have a good estimate for the rates of the server actions in our performance models, we used the same live experiment in [2] to derive some of the PEPA model rates. Moreover, we assumed that the rates of client actions to be 0.002 for actions that will be done by the client software and 0.0002 for actions to be done by the voter.

Table 1: Action Rates for *DRE-i* client.

Action	**Rate**
voteCryptogramsReq	0.00114
signTransReq	0.00082
voteCastMSG, voteVerifyMSG	0.00087
selectVote, castReply, verifyReply, reselect	0.0002
voteCastingComplete , voteVerificationComplete	0.002
reselectOrEndVotingRate	0.002
wait	0.0000519585

Table 2: Action Rates for *DRE-i* server.

Action	**Rate**
voteCryptogramsReply	0.00114
signTransReply	0.00082
voteCastAck, voteVerifyAck	0.00087

3.4 Throughput Analysis of Server Actions

The throughput of an action is defined as the average number of actions completed by the system during a unit of time (ms) [13]. In PEPA, we can calculate the average number of jobs waiting to be served by an action in the model. This number is called the population(mean queue length). Therefore, we can calculate the average number of valid voters and rogue voters waiting for each server's action. The throughput and population of model actions can be derived by PEPA Eclipse Plug-in immediately after solving the CTMC underlying the model or the ODE approximation of the CTMC. Because we are interested in investigating the impact of rogue clients' intervention in the DRE-i system, we need to evaluate the goodput and badput of each server action. Goodput of a server action expresses the throughput dedicated for the average number of legitimate clients waiting to be served by the server action(See formula **(1)**). The badput of a server action expresses the throughput dedicated for the average number of rogue clients waiting to be served by the server action(See formula **(2)**).

$$\textbf{Goodput} = \text{action throughput} \times \left(\frac{\text{number of valid voters}}{\text{total number of voters}}\right) \tag{1}$$

$$\textbf{Badput} = \text{action throughput} \times \left(\frac{\text{number of rogue voters}}{\text{total number of voters}}\right) \tag{2}$$

4 PEPA Models

We will construct the formal performance models for the typical behaviour of the DRE-i voting scheme using PEPA formalism similar to the PEPA model in [2]. However, in this model we will not model the voter behaviour because we will abstract the voter behaviour inside the voting client. Therefore, in this model we will have the voting client and the voting server components. Based on Figure 1 we define the formal performance model for the DRE-i voting scheme using PEPA language as follows:

4.1 DRE-i Server and Client

The system is composed of 30 *DRE-i* legitimate clients and one server. The system starts by receiving the request *voteCryptogramsReq* form the *DRE-i* client

and subsequently the *DRE-i* server replies with the action *voteCryptogramsReply* . The client and server continue the interactions as shown in the PEPA description of client and server collaboration below.

Voting client:

$\mathbf{DRE\_Client_0} \stackrel{def}{=} (\text{voteCryptogramsReq},r_{\text{voteCryptogramsReq}}).\text{DRE\_Client}_1$
$\mathbf{DRE\_Client_1} \stackrel{def}{=} (\text{voteCryptogramsReply},r_{\text{voteCryptogramsReply}}).\text{DRE\_Client}_2$
$\mathbf{DRE\_Client_2} \stackrel{def}{=} (\text{selectVote},r_{\text{selectVote}}).\text{DRE\_Client}_3$
$\mathbf{DRE\_Client_3} \stackrel{def}{=} (\text{signTranscriptReq},r_{\text{signTranscriptReq}}).\text{DRE\_Client}_4$
$\mathbf{DRE\_Client_4} \stackrel{def}{=} (\text{signTranscriptReply},r_{\text{signTranscriptReply}}).\text{DRE\_Client}_5$
$\mathbf{DRE\_Client_5} \stackrel{def}{=} (\text{castReply},r_{\text{castReply}}).\text{DRE\_Client}_6 + (\text{verifyReply},r_{\text{verifyReply}}).\text{DRE\_Client}_7$
$\mathbf{DRE\_Client_6} \stackrel{def}{=} (\text{castedVoteMSG},r_{\text{castedVoteMSG}}).\text{DRE\_Client}_8$
$\mathbf{DRE\_Client_8} \stackrel{def}{=} (\text{castedVoteAck},r_{\text{castedVoteAck}}).$
$(\text{voteCastingComplete},r_{\text{voteCastingComplete}}).(\text{wait},r_{\text{wait}}).\text{DRE\_Client}_0$
$\mathbf{DRE\_Client_7} \stackrel{def}{=} (\text{verifiedVoteMSG},r_{\text{verifiedVoteMSG}}).\text{DRE\_Client}_9$
$\mathbf{DRE\_Client_9} \stackrel{def}{=} (\text{verifiedVoteAck},r_{\text{verifiedVoteAck}}).$
$(\text{voteVerificationComplete},r_{\text{voteVerificationComplete}}).\text{DRE\_Client}_{10}$
$\mathbf{DRE\_Client_{10}} \stackrel{def}{=} (\text{reselectOrEndVoting},r_{\text{reselectOrEndVoting}}).\text{DRE\_Client}_{11}$
$\mathbf{DRE\_Client_{11}} \stackrel{def}{=} (\text{reselect},r_{\text{reselect}}).\text{DRE\_Client}_0 +$
$(\text{endVoting},r_{\text{endVoting}}).(\text{wait},r_{\text{wait}}).\text{DRE\_Client}_0$

Voting server:

$\mathbf{DRE\_SRV_0} \stackrel{def}{=} (\text{voteCryptogramsReply},r_{\text{voteCryptogramsReply}}).\text{DRE\_SRV}_0 +$
$(\text{signTranscriptReply},r_{\text{signTranscriptReply}}).\text{DRE\_SRV}_0 +$
$(\text{castedVoteAck},r_{\text{castedVoteAck}}).\text{DRE\_SRV}_0 + (\text{verifiedVoteAck},r_{\text{verifiedVoteAck}}).\text{DRE\_SRV}_0$

System equation:

$$(\mathbf{DRE\_Client_0}[i] \underset{\mathcal{L}_1}{\bowtie} \mathbf{DRE\_SRV_0}[j])$$

where i is the number of voters in the system, j is the number of e-voting servers, and

$\mathcal{L}_1 = \{ \text{voteCryptogramsReply,signTranscriptReply, castedVoteAck,verifiedVoteAck} \}$

4.2 RCA DRE-i clients

The *DRE-i* rouge client of type RCA starts interacting with the system by sending the request *voteCryptogramsReq* to the *DRE-i* server. The server replies with the action *voteCryptogramsReply* to end the collaboration and the *RCA DRE-i* rogue client goes back to the initial state $RCA\_DRE\_Client_0$.

RCA Voting client:

$\mathbf{RCA\_DRE\_Client_0} \stackrel{def}{=} (\text{voteCryptogramsReq},r_{\text{voteCryptogramsReq}}).\text{RCA\_DRE\_Client}_1$
$\mathbf{RCA\_DRE\_Client_1} \stackrel{def}{=} (\text{voteCryptogramsReply},r_{\text{voteCryptogramsReply}}).\text{RCA\_DRE\_Client}_2$
$\mathbf{RCA\_DRE\_Client_2} \stackrel{def}{=} (\text{rc\_wait},r_{\text{rc\_wait}}).\text{RCA\_DRE\_Client}_0$

System equation:

$$(\mathbf{DRE\_Client_0}[i] \bowtie \mathbf{RCA\_DRE\_Client_0}[k]) \underset{\mathcal{L}_1}{\bowtie} \mathbf{DRE\_SRV_0}[j])$$

where i is the number of voters in the system, j is the number of e-voting servers, k is the number of RCA clients, and

$\mathcal{L}_1 = \{ \text{voteCryptogramsReply,signTranscriptReply, castedVoteAck,verifiedVoteAck} \}$

4.3 RCB DRE-i clients

The *DRE-i* rouge client of type RCB starts interacting with the system by sending the request *voteCryptogramsReq* to the *DRE-i* server and successfully receiving the action *voteCryptogramsReply* from the server . In the next step, the RCB client select the candidate. Subsequently, the RCB client sends the request *signTranscriptReq* to the server and the server replies with the action *signTranscriptReply* to end the collaboration and the *RCB DRE-i* rogue client goes back to the initial state *RCB_DRE_ Client$_0$*.

RCB Voting client:

$\mathbf{RCB\_DRE\_Client_0} \stackrel{def}{=} (\text{voteCryptogramsReq},r_{\text{voteCryptogramsReq}}).\text{RCB\_DRE\_Client}_1$
$\mathbf{RCB\_DRE\_Client_1} \stackrel{def}{=} (\text{voteCryptogramsReply},r_{\text{voteCryptogramsReply}}).\text{RCB\_DRE\_Client}_2$
$\mathbf{RCB\_DRE\_Client_2} \stackrel{def}{=} (\text{selectVoteReq},r_{\text{selectVoteReq}}).\text{RCB\_DRE\_Client}_3$
$\mathbf{RCB\_DRE\_Client_3} \stackrel{def}{=} (\text{signTranscriptReq},r_{\text{signTranscriptReq}}).\text{RCB\_DRE\_Client}_4$
$\mathbf{DRE\_Client_4} \stackrel{def}{=} (\text{signTranscriptReply},r_{\text{signTranscriptReply}}).\text{DRE\_Client}_5$
$\mathbf{RCB\_DRE\_Client_5} \stackrel{def}{=} (\text{rc\_wait},r_{\text{rc\_wait}}).\text{RCB\_DRE\_Client}_0$

System equation:

$$((\mathbf{DRE\_Client_0}[i] \bowtie \mathbf{RCB\_DRE\_Client_0}[k]) \underset{\mathcal{L}_1}{\bowtie} \mathbf{DRE\_SRV_0}[j])$$

where i is the number of voters in the system, j is the number of e-voting servers, k is the number of RCB clients, and

$\mathcal{L}_1 = \{$ voteCryptogramsReply,signTranscriptReply, castedVoteAck,verifiedVoteAck $\}$

4.4 RCC DRE-i clients

In this type of rogue clients, the RCC *DRE-i* rouge client successfully collaborate with the server through sending the client actions *voteCryptogramsReq* , *signTransReq*, *voteCastMSG*, and *voteVerifyMSG* and receiving the server actions *voteCryptogramsReply*, *signTranscriptReply*, *castedVoteAck*, and *verifiedVoteAck*.

RCC Voting client:

$\mathbf{RCC\_DRE\_Client_0} \stackrel{def}{=} (\text{voteCryptogramsReq},r_{\text{voteCryptogramsReq}}).\text{RCC\_DRE\_Client}_1$
$\mathbf{RCC\_DRE\_Client_1} \stackrel{def}{=} (\text{voteCryptogramsReply},r_{\text{voteCryptogramsReply}}).\text{RCC\_DRE\_Client}_2$
$\mathbf{RCC\_DRE\_Client_2} \stackrel{def}{=} (\text{selectVoteReq},r_{\text{selectVoteReq}}).\text{RCC\_DRE\_Client}_3$
$\mathbf{RCC\_DRE\_Client_3} \stackrel{def}{=} (\text{signTranscriptReq},r_{\text{signTranscriptReq}}).\text{RCC\_DRE\_Client}_4$
$\mathbf{DRE\_Client_4} \stackrel{def}{=} (\text{signTranscriptReply},r_{\text{signTranscriptReply}}).\text{DRE\_Client}_5$
$\mathbf{RCC\_DRE\_Client_5} \stackrel{def}{=} (\text{castReply},r_{\text{castReply}}).\text{RCC\_DRE\_Client}_6 +$
$(\text{verifyReply},r_{\text{verifyReply}}).\text{RCC\_DRE\_Client}_7$
$\mathbf{RCC\_DRE\_Client_6} \stackrel{def}{=} (\text{castedVoteMSG},r_{\text{castedVoteMSG}}).\text{RCC\_DRE\_Client}_8$
$\mathbf{RCC\_DRE\_Client_8} \stackrel{def}{=} (\text{castedVoteAck},r_{\text{castedVoteAck}}).\text{RCC\_DRE\_Client}_9$
$\mathbf{RCB\_DRE\_Client_9} \stackrel{def}{=} (\text{voteCastingComplete},r_{\text{voteCastingComplete}}).$
$(\text{rc\_wait},r_{\text{rc\_wait}}).\text{RCB\_DRE\_Client}_0$
$\mathbf{RCC\_DRE\_Client_7} \stackrel{def}{=} (\text{verifiedVoteMSG},r_{\text{verifiedVoteMSG}}).\text{RCC\_DRE\_Client}_{10}$
$\mathbf{RCC\_DRE\_Client_{10}} \stackrel{def}{=} (\text{verifiedVoteAck},r_{\text{verifiedVoteAck}}).\text{RCC\_DRE\_Client}_{11}$
$\mathbf{RCC\_DRE\_Client_{11}} \stackrel{def}{=} (\text{voteVerificationComplete},r_{\text{voteVerificationComplete}}).$
$(\text{rc\_wait},r_{\text{rc\_wait}}).\text{RCB\_DRE\_Client}_0$

System equation:

$$((\mathbf{DRE\_Client_0}[i] \bowtie \mathbf{RCC\_DRE\_Client_0}[k]) \underset{\mathcal{L}_1}{\bowtie} \mathbf{DRE\_SRV_0}[j])$$

where i is the number of voters in the system, j is the number of e-voting servers, k is the number of RCC clients, and
$\mathcal{L}_1 = \{$ voteCryptogramsReply,signTranscriptReply, castedVoteAck,verifiedVoteAck $\}$

5 Results and Discussion

After constructing and testing the performance models using the PEPA Eclipse Plug-in tool, we used the ordinary differential equations technique [14] of the tool to evaluate the effect of the intervention of rogue clients on the DRE-i e-voting system. We evaluated the goodput and badput of the server actions for the three types of rogue clients.

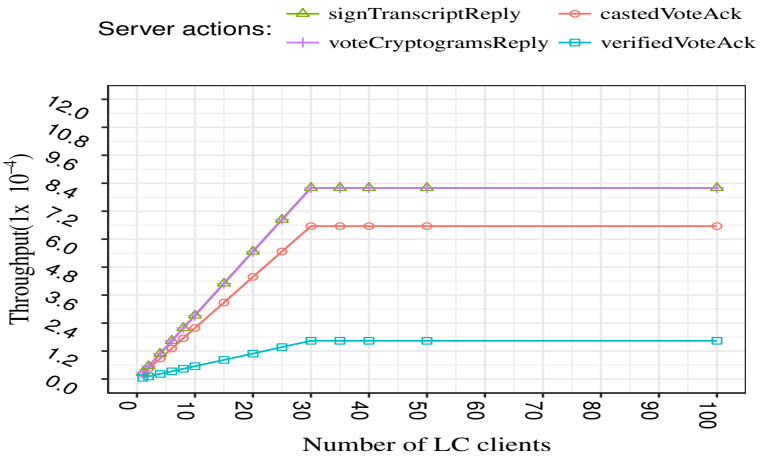

Fig. 2: Throughput for server actions.

First, we investigated the impact of the misbehaving voters on the DRE-i system that had one DRE-i server, thirty legitimate clients, and a varying number of rogue clients. The goodput and badput were analysed. Later, we fixed the number of rogue clients to be 100, the legitimate clients (LC) to be 30, and varied the number of DRE-i servers from one to eight. Before starting the evaluation of the impact of the misbehaving voters' intervention on the performance of the system, we evaluated the throughput of the system. The system had one server, a varying number of legitimate clients, and no rogue clients. We found out that the server action *signTranscriptReply* had reached its maximum rate of 0.000820 when there were 30 legitimate clients in the system. As a result, the *castedVoteAck* and *verifiedVoteAck* reached a maximum throughput of 0.000656 and 0.000164, and *voteCryptogramsReply* reached a maximum throughput of 0.000820 (See Fig. 2). Therefore, we used this configuration, the one server

and 30 legitimate clients, to evaluate the impact of the intervention of the three types of the misbehaving voters on the good throughput of the DRE-i system.

5.1 Goodput of Server Actions

In this section, we will show the effect of the three types of interventions of the misbehaving voters when they interact with one DRE-i server. Each intervention type has a different rate to complete one intervention with the DRE-i server. RCA will have the highest rate to complete one intervention, RCB will have a lower rate, and RCC will have the lowest rate.

Impact of RCA Intervention In the PEPA model of RCA, the rogue client will replay the request *voteCryptogramsReq* and wait for a reply from the server. The server will receive the request and reply with *voteCryptogramsReply* to end the interactions between the rouge client and the server. The impact of *voteCryptogramsReq* requests sent by rogue client of type RCA on the throughput of the DRE-i server actions is demonstrated in Fig. 3 and Fig. 4. The *voteCryptogramsReply* action has a maximum rate of 0.00114. The RCA clients make the *voteCryptogramsReply* action reach that maximum because RCA clients do not go through the *signTranscriptReply* action. The badput figure shows the increase of the server action throughput used by rogue clients when we gradually increase the number of rogue clients. Consequently, the goodput figure reveals that the more we add rogue clients to the system the less throughput will be dedicated for legitimate clients.

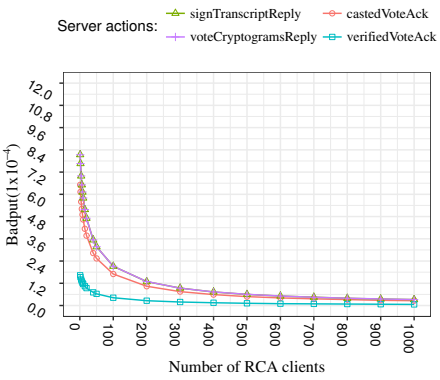

Fig. 3: Goodput for server actions.

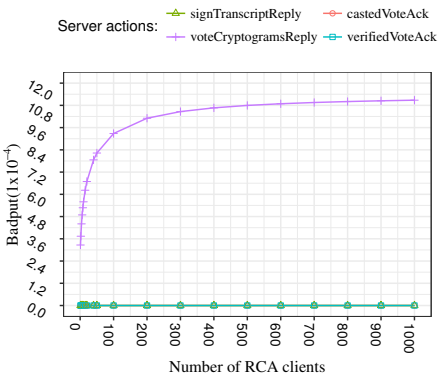

Fig. 4: Badput for server actions.

Impact of RCB Behaviour The impact of *voteCryptogramsReq* and *signTranscriptReq* requests sent by rogue client of type RCB on the throughput of the DRE-i server actions is demonstrated in Fig. 5 and Fig. 6. In this intervention

type, the rogue client needs to go through the server action *signTranscriptReply* which has the minimum rate among the rates of the server's actions. The rate of the server's action *signTranscriptReply* will slow down the intervention rate of RCB compared to the intervention rate of RCA. This explains why the *voteCryptogramsReply* action will not exceed the rate of 0.00082.

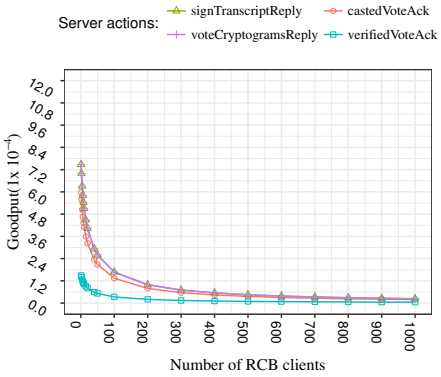

Fig. 5: Goodput for server actions.

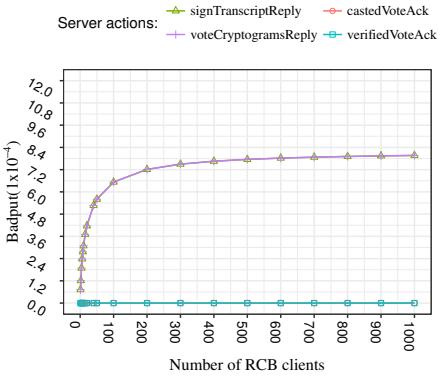

Fig. 6: Badput for server actions.

The badput and goodput in Fig. 5 and Fig. 6 show that the increase in the number of rogue clients in the system increases the badput of the server's actions. Consequently, this will make goodput of the server's actions decrease.

Impact of RCC Behaviour The impact of the badput of the RCC rogue client is shown in Fig. 7 and Fig. 8.

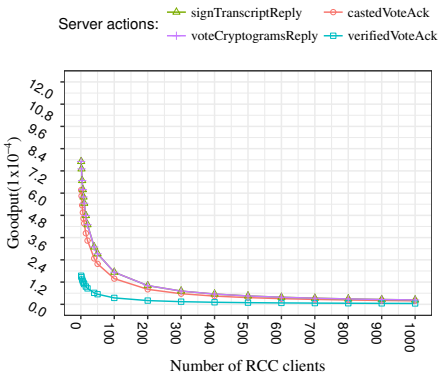

Fig. 7: Goodput for server actions.

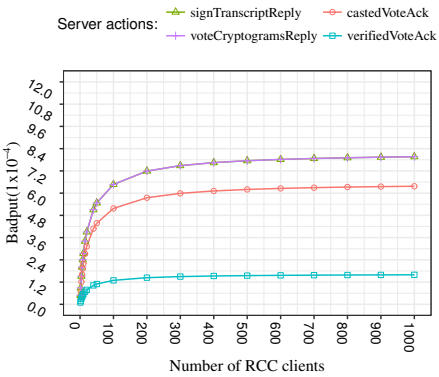

Fig. 8: Badput for server actions.

The badput and goodput of server actions' *voteCryptogramsReply* and *signTranscriptReply* in this type of intervention are similar to those in the RCB intervention because both RCB and RCC rogue clients need to go through the server action *signTranscriptReply*.

5.2 Scalability of Server's Goodput

After investigating the impact of the three types of the illegitimate interventions on one server, we have studied the goodput of server actions when there are more than one server. As shown in Fig. 9 and Fig. 10, the throughputs of the server actions *voteCryyptogramReply* and *castedVoteAck* are fixed at 0.0082 and 0.00065 when the system has 30 legitimate clients and varying number of servers.

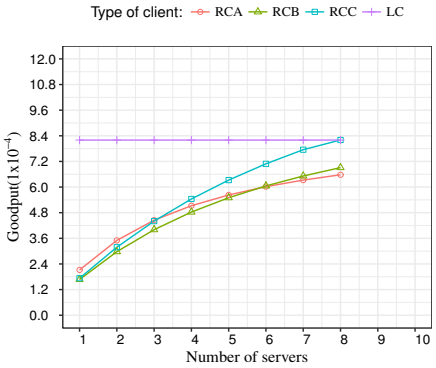

Fig. 9: Goodput for voteCryptogramsReply. LC=30, RCA=100, RCB=100, and RCC=100.

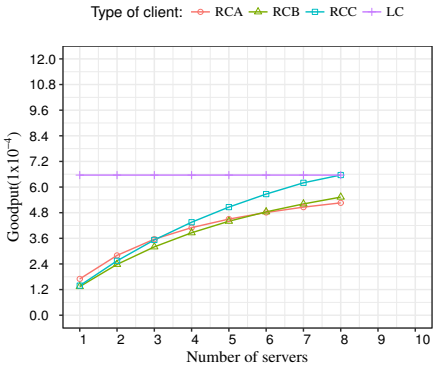

Fig. 10: Goodput for castedVoteAck. LC=30, RCA=100, RCB=100, and RCC=100.

The throughput for the two server's actions do not increase when we add more servers. The rate of 0.0082 for the action *signTranscriptReply* (one server) is enough to provide the 30 legitimate clients with required resources. In the case when there are one or two servers in the DRE-i system, the goodput of the server action *voteCryyptogramReply* when the system interacts with 100 RCA clients is better than the goodput of the server action *voteCryyptogramReply* when the system interacts with 100 RCC clients. However, when there are five servers, we notice the contrary. This is because the RCA rogue client has a higher intensity of actions with the DRE-i server than the RCB or RCC rogue client has. The RCB and RCC rogue clients face a bottleneck at the server's action *signTranscriptReply* when they interact with the system. However, when we increase the number of servers, we alleviate the bottleneck in the action *signTranscriptReply*. Therefore, the rogue clients RCB and RCC, and the 30 legitimate clients get more throughput from the server's action *signTranscriptReply*. So, the DRE-i

system starts having a better goodput when it has interventions from RCB or RCC rogue clients compared to the goodput it will have when it has interventions from RCA. Moreover, the evaluation of the performance models of the DRE-i system with misbehaving voters shows that adding more servers (up to seven servers) do not make the goodput of the servers' actions reaches the throughput of the DRE-i servers when the system has no misbehaving voters.

6 Conclusion

In this paper, by using PEPA, we presented the performance models of three misbehaving voters when using the large scale and secure DRE-i e-voting system. The constructed performance models captured the high-level interactions between the DRE-i e-voting system, the valid voters, and the misbehaving voters. The evaluation of throughput of the main DRE-i server's actions clearly shows the impact of the interaction of misbehaving voters with the e-voting system. The goodput of server actions went down when added more rogue clients to the system.

The evaluation of the effect of misbehaving voters on the DRE-i e-voting system can be extended to include the analysis of the response time that will be observed by legitimate voters when they cast their votes. Furthermore, the countermeasures to reduce the effect of misbehaving voters on the performance of the DRE-i voting system is an interesting area to be investigated using the formal performance formalism like PEPA.

References

1. Adida, B.: Helios: Web-based open-audit voting. In: USENIX security symposium. vol. 17, pp. 335–348 (2008)
2. Alotaibi, M., Thomas, N.: Performance evaluation of a secure and scalable e-voting scheme using PEPA. In: Workshop on New Frontiers in Quantitative Methods in Informatics. pp. 35–48. Springer (2017)
3. Benaloh, J., Vaudenay, S., Quisquater, J.J.: Final report of IACR electronic voting committee. international association for cryptologic research (2010)
4. Bradley, J.T., Gilmore, S.T.: Stochastic simulation methods applied to a secure electronic voting model. Electronic Notes in Theoretical Computer Science **151**(3), 5–25 (2006)
5. Brightwell, I., Cucurull, J., Galindo, D., Guasch, S.: An overview of the iVote 2015 voting system (2015)
6. Cortier, V., Smyth, B.: Attacking and fixing helios: An analysis of ballot secrecy. Journal of Computer Security **21**(1), 89–148 (2013)
7. Estehghari, S., Desmedt, Y.: Exploiting the client vulnerabilities in Internet e-voting systems: Hacking Helios 2.0 as an example. EVT/WOTE **10**, 1–9 (2010)
8. Fujioka, A., Okamoto, T., Ohta, K.: A practical secret voting scheme for large scale elections. In: International Workshop on the Theory and Application of Cryptographic Techniques. Springer (1992)

9. Gilmore, S., Tribastone, M.: Evaluating the scalability of a web service-based distributed e-learning and course management system. In: International Workshop on Web Services and Formal Methods. pp. 214–226. Springer (2006)
10. Gritzalis, D.A.: Principles and requirements for a secure e-voting system. Computers & Security **21**(6), 539–556 (2002)
11. Hao, F., Kreeger, M., Randell, B., Clarke, D., Shahandashti, S., Lee, P.J.: Every vote counts: Ensuring integrity in large-scale electronic voting. In: 2014 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE 14). vol. 2, pp. 1–25 (2014)
12. Hao, F., Ryan, P.Y.: Real-world Electronic Voting: Design, Analysis and Deployment. CRC Press (2016)
13. Hillston, J.J.: A compositional approach to performance modelling. Distinguished dissertations in computer science, Cambridge University Press, Cambridge ; New York (1996)
14. Hillston, J.: Fluid flow approximation of PEPA models. In: Second International Conference on the Quantitative Evaluation of Systems (QEST'05). IEEE (2005)
15. Kohno, T., Stubblefield, A., Rubin, A.D., Wallach, D.S.: Analysis of an electronic voting system. In: Security and Privacy, 2004. Proceedings. 2004 IEEE Symposium on. pp. 27–40. IEEE (2004)
16. Lee, P.H.J., Shahandashti, S.F.: Theoretical attacks on E2E voting systems. Real-World Electronic Voting: Design, Analysis and Deployment p. 219 (2016)
17. Madise, Ü., Martens, T.: E-voting in Estonia 2005. the first practice of country-wide binding Internet voting in the world. Electronic voting **86**
18. Smyth, B., Cortier, V.: A note on replay attacks that violate privacy in electronic voting schemes. Ph.D. thesis, INRIA (2011)
19. Sudhodanan, A., Carbone, R., Compagna, L., Dolgin, N., Armando, A., Morelli, U.: Large-scale analysis & detection of authentication cross-site request forgeries. In: Security and Privacy (EuroS&P), 2017 IEEE European Symposium on. pp. 350–365. IEEE (2017)
20. Thomas, N.: Performability of a secure electronic voting algorithm. Electronic Notes in Theoretical Computer Science (2005)
21. Tribastone, M., Duguid, A., Gilmore, S.: The PEPA eclipse plugin. ACM SIGMETRICS Performance Evaluation Review **36**(4), 28–33 (2009)
22. Wang, H., Laurenson, D.I., Hillston, J.: Evaluation of RSVP and mobility-aware RSVP using performance evaluation process algebra. In: Communications, 2008. ICC'08. IEEE International Conference on. pp. 192–197. IEEE (2008)
23. Yasinac, A., Wagner, D., Bishop, M., Baker, T., de Medeiros, B., Tyson, G., Shamos, M., Burmester, M.: Software review and security analysis of the ES&S iVotronic 8.0. 1.2 voting machine firmware. Tech. rep., Security and Assurance in Information Technology Laboratory, Florida State University (2007)